1-21-05

AF ZFW

PTO/SB/21 (09-04)

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Application Number	09/991,379
Filing Date	November 15, 2001
First Named Inventor	Mascavage, John Joseph
Art Unit	3628
Examiner Name	Siegfried E. Chencinski
Total Number of Pages in This Submission: 33	Attorney Docket Number: **020375-002710US**

ENCLOSURES *(Check all that apply)*

- ☐ Fee Transmittal Form
  - ☐ Fee Attached
- ☐ Amendment/Reply
  - ☐ After Final
  - ☐ Affidavits/declaration(s)
- ☐ Extension of Time Request
- ☐ Express Abandonment Request
- ☐ Information Disclosure Statement
- ☐ Certified Copy of Priority Document(s)
- ☐ Reply to Missing Parts/ Incomplete Application
  - ☐ Reply to Missing Parts under 37 CFR 1.52 or 1.53
- ☐ Drawing(s)
- ☐ Licensing-related Papers
- ☐ Petition
- ☐ Petition to Convert to a Provisional Application
- ☐ Power of Attorney, Revocation Change of Correspondence Address
- ☐ Terminal Disclaimer
- ☐ Request for Refund
- ☐ CD, Number of CD(s) _____
  - ☐ Landscape Table on CD
- ☐ After Allowance Communication to TC
- ☐ Appeal Communication to Board of Appeals and Interferences
- ☒ **Appeal Communication to TC (Appeal Brief)**
- ☐ Proprietary Information
- ☐ Status Letter
- ☒ **Other Enclosure(s)** (please identify below):
  - Return Postcard
  - Appendices A, B and C

Remarks: The Commissioner is authorized to charge any additional fees to Deposit Account 20-1430.

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm Name	Townsend and Townsend and Crew LLP		
Signature			
Printed name	Thomas D. Franklin		
Date	January 18, 2005	Reg. No.	43,616

CERTIFICATE OF TRANSMISSION/MAILING

Express Mail Label: EV 291388424 US

I hereby certify that this correspondence is being deposited with the United States Postal Service with "Express Mail Post Office to Address" service under 37 CFR 1.10 on this date **January 18, 2005** and is addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below.

Signature			
Typed or printed name	Cindy Bennett	Date	January 18, 2005

"Express Mail" Label No. EV 291388424 US
Date of Deposit January 18, 2005

PATENT
Attorney Docket No.: 020375-002710US

I hereby certify that this is being deposited with the United States Postal Service "Express Mail Post Office to Address" service under 37 CFR 1.10 on the date indicated above and is addressed to:

MAIL STOP: APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

By: Cindy Bennett

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re application of:

John Joseph Mascavage III, et al.

Application No.: 09/991,379

Filed: November 15, 2001

For: ONLINE PURCHASING METHOD

Customer No. 20350

Confirmation No.: 2669

Examiner: Chencinski, Siegfried E.

Technology Center/Art Unit: 3628

APPELLANT BRIEF UNDER 37 CFR §41.37

MAIL STOP: APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

Appellant offers this Brief further to the Notice of Appeal mailed on November 15, 2004.

1. Real Parties in Interest

Western Union Financial Services, Inc. is the real party in interest of the above-identified application. Western Union Financial Services, Inc. is a subsidiary of First Data Corp., a publicly-traded entity.

2. Related Appeals and Interferences

No other appeals or interferences are known that will directly affect, are directly affected by, or have a bearing on the Board decision in this appeal.

3. Status of Claims

Claims 1 – 20 are currently pending in the application. All pending claims stand finally rejected pursuant to a final Office Action mailed July 15, 2004 and further explained in an Advisory Action mailed November 4, 2004. The rejections of claims 1 – 20 are believed to be improper and are the subject of this appeal.

4. Status of Amendments

The claims have never been amended. This appeal is based upon the state of the claims filed in the application on November 15, 2001.

5. Summary of Claimed Subject Matter

According to the embodiment of claim 1, a process for authorizing an online purchase between a customer and a vendor site is disclosed. Application, Figs. 6, 11 and 12. In one step, transaction information is received from the vendor site. Id., Fig. 12, step 1216; page 19, lines 24-28. A new web browser window is automatically opened for the customer. Id., Fig. 11, step 1124; page 18, lines 20-24. A transaction amount is presented in a new browser window. Id., Fig. 12, step 1220; Fig. 11, step 1144; page 18, line 33 through page 19, line 1. The customer is capable of assenting to the transaction amount through interaction with the new web browser window. Id., Fig. 12, step 1224; page 19 lines 4-5 and line 33. Authorization is received from the customer of a debit for the transaction amount to cover the online purchase. Id., Fig. 12, step 1224; page 19, lines 3-4 and line 33. The vendor site is notified of the authorization. Id., Fig. 12, step 1232; page 20, lines 6-7.

In the embodiment of claim 10, a method for checking-out from an online purchase by a customer from a merchant system is disclosed. Application, Figs. 6, 11 and 12. In

one step, transaction information is received from the merchant system. Id., Fig. 12, step 1216; page 19, lines 24-28. Automatically, a window is opened that is viewable by the customer, where the window is formulated by a funds transfer system at a network location away from the merchant system. Id., Fig. 11, step 1124; page 18, lines 20-24. A transaction amount is presented in the window, whereby the customer can assent to the transaction amount by interacting with the window. Id., Fig. 12, step 1220; Fig. 11, step 1144; page 18, line 33 through page 19, line 1. Authorization is received from the customer of a debit for the transaction amount, where the debit corresponds to the online purchase. Id., Fig. 12, step 1224; page 19, lines 3-4 and line 33. The merchant system is notified of the authorization. Id., Fig. 12, step 1232; page 20, lines 6-7.

The embodiment of claim 17 discloses a method for checking-out from an online purchase by a customer from a merchant system. Application, Figs. 6, 11 and 12. In one step, account information is received from the customer corresponding to an account available for debits by the funds transfer system. Id., Fig. 12, step 1216; page 19, lines 24-28. Automatically, opening a window that is viewable by the customer, wherein the window is formulated by the funds transfer system at a site away from the merchant system. Id., Fig. 11, step 1124; page 18, lines 20-24. A transaction amount is presented in the window, whereby the customer can assent to the transaction amount by interacting with the window. Id., Fig. 12, step 1224; page 19 lines 4-5 and line 33. Authorization is received from the customer of a debit for the transaction amount, wherein the debit corresponds to the online purchase. Id., Fig. 12, step 1224; page 19, lines 3-4 and line 33. The merchant system is notified of the authorization. Id., Fig. 12, step 1232; page 20, lines 6-7.

6. Grounds of Rejection Presented for Review

A. The priority claim to U.S. Patent Application No. 09/516,209 ("Parent Application") is rejected under 35 U.S.C. §112, first paragraph with regard to the slightly different limitations: "automatically opening a new web browser window for the customer" of

claim 1, and "automatically opening a window that is viewable by the customer" of claims 10 and 17.

B. Claims 1-7, 9-15 and 17-20 stand rejected under 35 U.S.C. §103(a) as being unpatentable over the cited portions of U.S. Patent No. 5,899,980 to Wilf et al. (hereinafter "Wilf") in view of the cited portions of U.S. PreGrant Publication No. 2002/0055909 to Fung (hereinafter "Fung").

C. Claims 8 and 16 are also under 35 U.S.C. §103(a) as being unpatentable over Wilf in view of Fung and further in view of the cited portions of U.S. Patent No. 5,920,847 to Kolling et al. (hereinafter "Kolling").

7. Argument

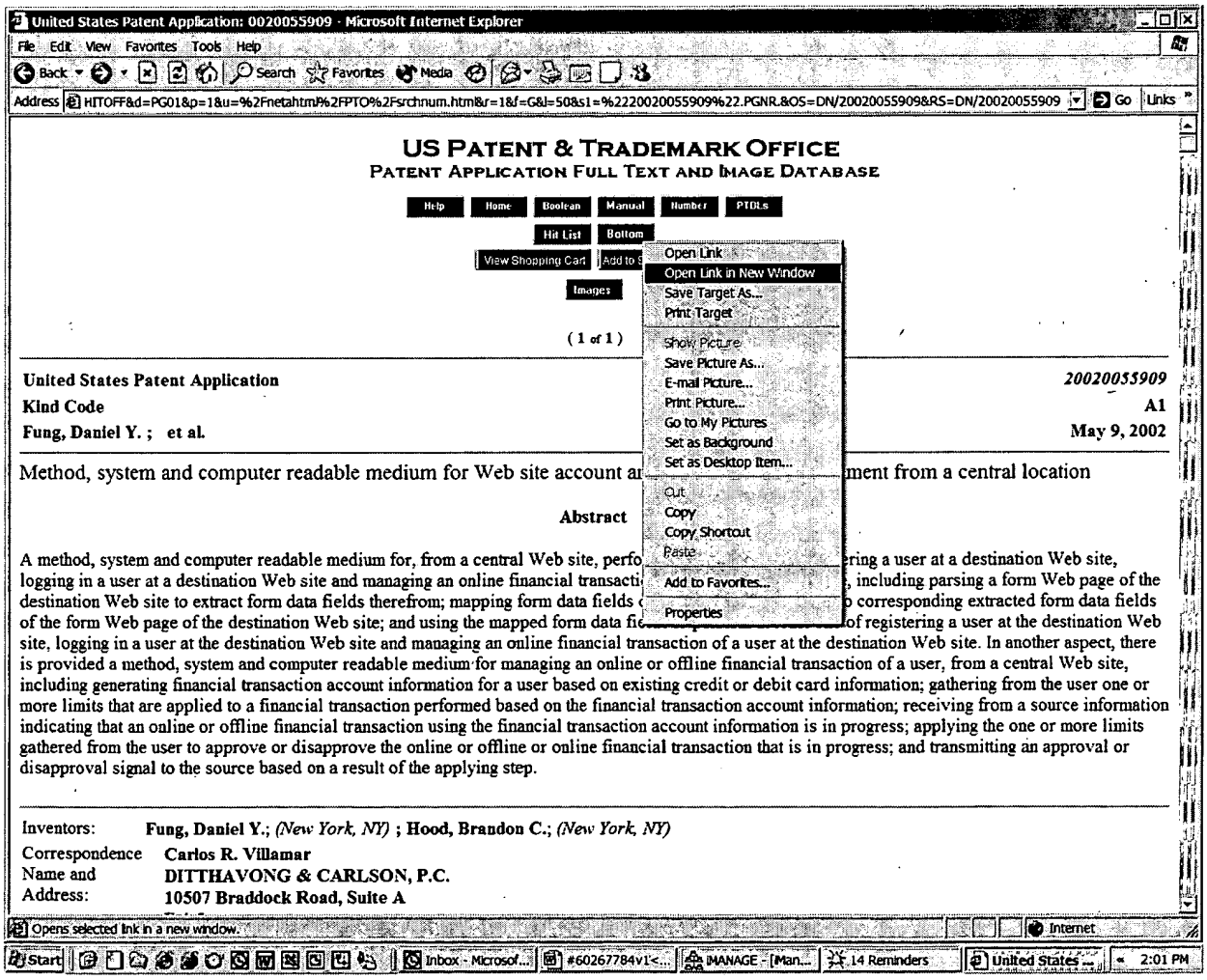
A. Whether the Parent Application Supports a Limitation in Claims 1, 10 and 17

Appellant's claim to the filing date benefit is rejected as not in compliance under 35 U.S.C. §112, first paragraph. The Examiner apparently takes the position that the following limitations are not supported in the Parent Application, specifically, "automatically opening a new web browser window for the customer" of claim 1 and "automatically opening a window that is viewable by the customer" of claims 10 and 17. Taken in context, the Parent Application specifies a pop-up window that displays a web page using an Internet browser. Id., page 19, line 23 through page 20, line 24. This portion of the application makes clear that a web browser is being used and the term "pop-up window" is notoriously well known. Appellant believes there is ample support for these claim limitations in the Parent Application, which would validate the priority claim and prevent use of Fung as prior art.

One of ordinary skill in the art programming in the HTML or JAVA™ language, used for web browsers, near the Feb. 2000 priority date would have known very well what a "pop-up window" means. The term is consistent to usage today and familiar to all web browser users. Pop-up windows with various ads have become notoriously well known. A whole industry has been created to produce software to block these pop-up windows in web browsers.

Indeed, the recently released version of Internet Explorer™ provided in service pack 2 of Windows XP™ even includes a pop-up window blocker.

The undersigned googled the term "pop-up window" the other day and found there were over 2 million uses of the term. Two examples attached as Appendixes B and C that were first published around the priority date of the Parent Application. These two examples demonstrate that "pop-up window" is a specific term of art meaning a new web browser window that opened without a direct request from the user. As one of ordinary skill appreciates along with any web browser user, a pop-up window automatically opens in a new web browser window. Windows can be manually opened, but pop-up windows open automatically.



The Appellant explains the difference between automatically and manually opening a new window. HTML links in web pages can be manually opened in a new web browser window. This is accomplished by a right-click on the link that brings up a context-sensitive menu, which is shown in the sole figure. Then, the user can manually select to open the link in the same window (i.e., the first option on the context-sensitive menu) or in a new window (i.e., the second option in the context-sensitive menu). This process would be described as the user manually opening a new web browser window. This is the opposite situation as claimed, but provides context for an understanding of what automatically opening a new web browser means to one of ordinary skill in the web programming art.

Well within the ordinary skill in the art, HTML programming language is used to design the web pages that are read by web browsers. When a link is embedded into a web page, the designer can either present the linked object in the same window or a new window. For example, to load the object "patent.html" in the same window with a textual link called "Patent," the HTML command would be: <A HREF="patent.html" TARGET=_top>Patent</A>. Alternatively, the object "patent.html" could be loaded in a new pop-up window using the command: <A HREF="patent.html" TARGET=_blank>Patent</A>. With either of these commands, the link would appear in the text as "Patent" such that a user cannot discern which will happen before selecting the link with a click.

The user cannot know by viewing the rendered web page if an activated link will put the linked information in the same window or a new window. The user activates the link and either action could occur based upon how the HTML hosted on the server of the web page is coded. That is to say, the user cannot manually affect the opening of a new web browser window after activating the link if the HTML is coded to open a new window. Unless the user performs the manual steps with the context-sensitive menu before activation of the link, the user has no control over whether the linked information appears in the same window or a new window. The server of the web page receives the click and automatically decides (based upon the HTML) if a new web browser window is opened or not. That is to say, the server controls the "automatically opening a new web browser window for the customer." Application, claim 1, line 4.

Appellant believes use of the term "pop-up window" in the context of web browsing, fully supports the claim limitations questioned, namely automatically opening a new window or new web browser window. One of ordinary skill in the art knows that a pop-up web browser window involves automatically opening that window. Since the user cannot know whether activating the link will bring the content into the current window or open a new window, the opening of a new window is properly termed as "automatic." Any interpretation to the contrary is believed unreasonable. Reconsideration is respectfully requested.

B. Whether claims 1-7, 9-15 and 17-20 are made obvious by a combination of Wilf & Fung

The Office Action has rejected claims 1-7, 9-15 and 17-20 under 35 U.S.C. §103(a) as being unpatentable over the cited portions of U.S. Patent No. 5,899,980 to Wilf et al. (hereinafter "Wilf") in view of the cited portions of U.S. PreGrant Publication No. 2002/0055909 to Fung (hereinafter "Fung"). The patent office (the "Office") is charged with putting forth a *prima facie* showing of obviousness. Applicants believe a *prima facie* case of obviousness has not been properly set forth in the final Office Action or Advisory Action. The basic test is excerpted below:

> "To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations.
>
> The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, not in the applicant's disclosure." MPEP §2143, Original Eighth Edition, August, 2001, Latest Revision May 2004.

As an initial matter, Fung is not prior art with regard to the automatically opening of a window (claims 10 and 17) or web browser window (claim 1), because of support in the

Parent Application as explained in the preceding section. Further, there are other limitations missing from Fung and proper motivation for the suggested combination is lacking.

Missing Limitations

Fung is cited for the teaching automatically opening a new window. The present claims generally require opening a new window and presenting a transaction amount in that window. Appellant concedes that pop-up windows are known (e.g., see Appendixes B and C), but Fung uses a pop-up window in a completely different way. In Fung, one window is used for the login process before the interaction with the site is performed in another window. Fung, paragraph 0056, lines 16-20. The site interaction can be performed "in a new web browser window or in the user's central Web site start page **206**." Id. After login, the user in Fung would proceed to a new pop-up window or an existing start page to interact with the site. There is no mention in Fung that the new or existing window would have the transaction amount. Fung can only be relied upon to teach a pop-up window and not the claimed invention that requires a transaction amount be presented in that new window.

Motive to Combine

Appellant believes motivation for the specific combination of elements in the cited references is lacking. The Advisory Action cites a reference for the motivation for the first time. The Advisory Action, at section 2d, cites a general motive for the combination, but curiously uses Fung for this motive. The Advisory Action is not clear whether Fung (i.e., U.S. PreGrant Publication No. 2002/0055909, filed June 14, 2001, to Fung) is the basis for this motive or if U.S. Provisional Patent 60/186,303, filed March 1, 2000 ("Fung Parent") is the basis for this motive. The cite in the Advisory Action gives a paragraph number for this motive so Fung is presumed to be the source, although Fung, paragraph [0006], lines 5-12, does not fully support the cited motive. A quick review of the Fung Parent does not reveal this motive either. Regardless, Fung and the Fung Parent are not prior art.

To be a valid cite to a motive, the Examiner must find a motive present in a prior art reference. The apparent source for the motive cited in the Advisory Action for the first time is Fung, which was filed on June 14, 2001. Any reliance on this reference for a motive is

believed unreasonable. A cite to a prior art reference or official notice must be used to support a motive to combine these references. Reconsideration is respectfully requested.

B. Whether claims 8 and 16 are made obvious by a combination of Wilf, Fung & Kolling

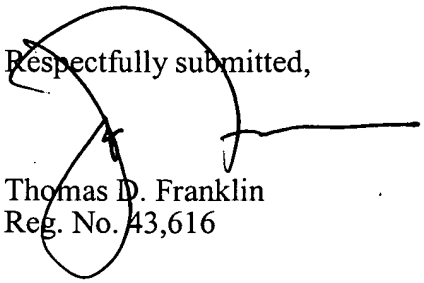
Claims 8 and 16 are rejected under 35 U.S.C. §103(a) as being unpatentable over Wilf in view of Fung and further in view of Kolling. Besides the missing limitations and improper motive to combine Wilf and Fung discussed in the preceding section, the motive to add Kolling to any combination is lacking. The final Office Action says the motive is to "protect a vendor from undue delay in verifying such a transaction." Final Office Action, page 5, section 3, last sentence. No cite is given for this motive and the final Office Action does not indicate that Official Notice is the source for this motive. Should the Office clarify the situation to indicate that Official Notice is the basis for this motive, an express showing of documentary proof is hereby requested.

8. Conclusion

Please deduct the requisite fee, pursuant to 37 C.F.R. §1.17(c), of $500.00 from deposit account 20-1430 and any additional fees associated that may be due in association with the filing of this Brief.

If for any reason the Office believes a telephone conference would in any way expedite resolution of the issues raised in this appeal, the Office is invited to telephone the undersigned attorney at (303) 571-4000.

Respectfully submitted,



Thomas D. Franklin
Reg. No. 43,616

TOWNSEND and TOWNSEND and CREW LLP
Two Embarcadero Center, 8th Floor
San Francisco, California 94111-3834
Tel: 303-571-4000
Fax: 415-576-0300

60399431 v1

APPENDIX A

The claims pending in the application are as follows:

1. (Original) A method for authorizing an online purchase between a customer and a vendor site, the method comprising steps of:

receiving transaction information from the vendor site;

automatically opening a new web browser window for the customer;

presenting a transaction amount in the new web browser window, whereby the customer can assent to the transaction amount through interaction with the new web browser window;

receiving authorization from the customer of a debit for the transaction amount, wherein the debit corresponds to the online purchase; and

notifying the vendor site of authorization.

2. (Original) The method for authorizing the online purchase between the customer and the vendor site as recited in claim 1, wherein the new web browser window points away from the vendor site.

3. (Original) The method for authorizing the online purchase between the customer and the vendor site as recited in claim 1, further comprising a step of receiving account information from the customer corresponding to an account authorized for the debit.

4. (Original) The method for authorizing the online purchase between the customer and the vendor site as recited in claim 1, wherein the new web browser window overlays an existing web browser window of the vendor site.

5. (Original) The method for authorizing the online purchase between the customer and the vendor site as recited in claim 1, wherein the receiving transaction information step triggers the automatically opening step.

6. (Original) The method for authorizing the online purchase between the customer and the vendor site as recited in claim 1, further comprising a step of transferring payment to an account associated with the vendor site after authorization is received.

7. (Original) The method for authorizing the online purchase between the customer and the vendor site as recited in claim 1, further comprising a step of presenting a message to the customer in the new web browser window indicating at least one of the following:

that authorization was canceled by the customer;

that authorization was rejected by a funds transfer system; and

that authorization completed normally.

8. (Original) The method for authorizing the online purchase between the customer and the vendor site as recited in claim 1, wherein the notifying step comprises a step of determining that a notification message was not received by the vendor site within a predetermined time period.

9. (Original) A computer-readable medium having computer-executable instructions for performing the computer-implementable method for authorizing the online purchase between the customer and the vendor site of claim 1.

10. (Original) A method for checking-out from an online purchase by a customer from a merchant system, the method comprising steps of:

receiving transaction information from the merchant system;

automatically opening a window that is viewable by the customer, wherein the window is formulated by a funds transfer system at a network location away from the merchant system;

presenting a transaction amount in the window, whereby the customer can assent to the transaction amount by interacting with the window;

receiving authorization from the customer of a debit for the transaction amount, wherein the debit corresponds to the online purchase; and

notifying the merchant system of authorization.

11. (Original) The method for checking-out from the online purchase by the customer from the merchant as recited in claim 10, further comprising a step of receiving account information from the customer corresponding to an account available for debits by the funds transfer system.

12. (Original) The method for checking-out from the online purchase by the customer from the merchant system as recited in claim 10, wherein the window overlays an existing web browser window of a web site associated with the merchant system.

13. (Original) The method for checking-out from the online purchase by the customer from the merchant system as recited in claim 10, wherein the receiving transaction information step triggers the automatically opening step.

14. (Original) The method for checking-out from the online purchase by the customer from the merchant system as recited in claim 10, further comprising a step of transferring payment to an account associated with the merchant system after authorization is received.

15. (Original) The method for checking-out from the online purchase by the customer from the merchant system as recited in claim 10, further comprising a step of presenting a message to the customer in another window indicating at least one of the following:

that authorization was canceled by the customer;

that authorization was rejected by the funds transfer system; and

that authorization completed normally.

16. (Original) The method for checking-out from the online purchase by the customer from the merchant system as recited in claim 10, wherein the notifying step

comprises a step of determining that a notification message was not received by the merchant system within a predetermined time period.

17. (Original) A method for checking-out from an online purchase by a customer from a merchant system, the method comprising steps of:

receiving account information from the customer corresponding to an account available for debits by the funds transfer system;

automatically opening a window that is viewable by the customer, wherein the window is formulated by the funds transfer system at a site away from the merchant system;

presenting a transaction amount in the window, whereby the customer can assent to the transaction amount by interacting with the window;

receiving authorization from the customer of a debit for the transaction amount, wherein the debit corresponds to the online purchase; and

notifying the merchant system of authorization.

18. (Original) The method for checking-out from the online purchase by the customer from the merchant system as recited in claim 17, wherein the account information is received through the window.

19. (Original) The method for checking-out from the online purchase by the customer from the merchant system as recited in claim 17, further comprising a step of receiving transaction information from the merchant system.

20. (Original) The method for checking-out from the online purchase by the customer from the merchant system as recited in claim 17, further comprising a step of transferring payment to an account associated with the merchant system after authorization is received.

Published on: Monday 9th November 1998 ***By:*** Martin Webb

Appendix B

Introduction to JavaScript Pop-up Windows

For most people, the main browser window is the only one they ever use or need. However, it is possible for another window to be opened up to allow you to either view another page while retaining the existing page in the main window, or to open up a remote window that can control or be controlled by the main browser window.

Another browser window can be simply opened using HTML and the ***TARGET*** attribute, either in a link or on a form:

```
<a href="testpage.htm" target="anotherWindowName">Open new window</a>
```

or:

```
<form action="testpage.htm" target="anotherWindowName">
<input type="submit" value="Open new window">
</form>
```

As long as the ***TARGET*** attribute specifies a window name that is not already open, a new window will be created mimicing the existing window. If the window already exists, then the contents will be changed instead.

The problem with this approach is that apart from the window name there is nothing in common between the two windows. You can change the location of the new window using a similar link for form target - but that's it. Whereas using JavaScript and the ***window.open()*** method, it is possible to control the look and feel of the new window, to have control over its contents, and also to be controlled by the new window.

Window open syntax

The syntax of the ***open()*** method is fairly straight forward:

```
windowHandle = window.open([URL [, windowName [, features]]])
```

In addition to this, MSIE4 has a fourth parameter:

```
windowHandle = window.open([URL [, windowName [, features [, replace]]]])
```

windowHandle

The result of the ***open()*** method is returned and held in the variable to the left of the assignment operator (=). This is a reference or handle to the newly opened window. It can be used to close the window, and to interrogate and alter the windows properties - more on this later.

URL

This is the relative or absolute URL of the file to be loaded into the pop-up window, for example:

```
<script language="JavaScript"><!--
var myWindow = window.open('http://www.irt.org/');
//--></script>

<a href="javascript:window.open('nextpage.htm')">Open Next Page</a>

<form>
<input type="button" onClick="window.open('picture.gif')">
</form>
```

If no URL is specified, a new window with ***about:blank*** will be displayed.

windowName

This is the target name of the new window. So you could load another document into it with:

```
<script language="JavaScript"><!--
window.open('testpage.htm','myWindow');
//--></script>

<a href="filename.html" target="myWindow">load file into popup window</a>
```

Features

The third parameter can hold a large selection of name-value pairs:

NN2.0+	Value	Description
directories	boolean	Controls the standard browser directory buttons
height	numeric	Specifies the height of the window in pixels
location	boolean	Controls the Location entry field
menubar	boolean	Controls the menu at the top of the window

resizable	boolean	Controls the ability to resize the window
scrollbars	boolean	Controls the horizontal and vertical scrollbars
status	boolean	Controls the status bar at the bottom of the window
toolbar	boolean	Controls the standard browser toolbar
width	numeric	Specifies the width of the window in pixels

If you don't specify the ***width*** or ***height*** values, then Netscape browsers will mimic the size of the current window, if you don't specify *any* attributes then Netscape will mimic all the features of the current window:

```
window.open('testpage.htm','myExample2');
```

Test Example 1

Note all the name-value pairs must be separated by commas, but must *not* have spaces between them. For example, the following is **incorrect:**

```
window.open('testpage.htm','myExample2','location=yes,  menubar=yes');
```

Whereas the following, which does not have spaces, is correct:

```
window.open('testpage.htm','myExample3','location=yes,menubar=yes');
```

Test Example 3

The values for the boolean parameters may be ***0*** or ***1*** or ***yes*** or ***no***. If a particular boolean parameter is not specified then its value defaults to ***no***. For example, to open a window without directory buttons, location field, menubar, scrollbars, status or toolbar and not resizable use:

```
window.open('testpage.htm','myExample4','width=200,height=200');
```

Test Example 4

Again, if you don't specify the width and height then the page mimics the current window.

However, to create a window with all the window features you need to explicitly include them:

```
window.open('testpage.htm','myExample5','width=200,height=200,directories=yes,location=yes,menubar=ye
```

Test Example 5

Replace
The fourth parameter is an optional boolean value, either true or false, which in MSIE4.0+ replaces the current history entry with the new page being loaded (i.e., the last location in the browsers history object is overwritten by the new location).

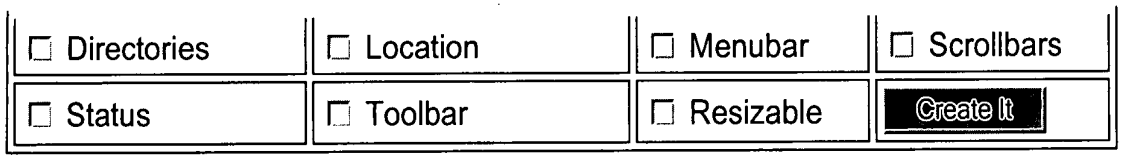
Stringing it together

So now that we know how to create a window with any features we desire, how can we use JavaScript to control the features we want? The first three parameters are all strings; JavaScript can manipulate strings just as well as any other programming language.

```
<script language="JavaScript"><!--
function createWindow(what) {
    var URL = what.URL.value;

    var windowName = what.windowName.value;

    var features =
        'width='        + what.width.value +
        ',height='      + what.height.value +
        ',directories=' + (what.directories.checked - 0) +
        ',location='    + (what.location.checked - 0) +
        ',menubar='     + (what.menubar.checked - 0) +
        ',scrollbars='  + (what.scrollbars.checked - 0) +
        ',status='      + (what.status.checked - 0) +
        ',toolbar='     + (what.toolbar.checked - 0) +
        ',resizable='   + (what.resizable.checked - 0);

    window.open (URL, windowName, features);
}
//--></script>

<form>
URL <input type="text" name="URL" value="testpage.htm">
Name <input type="text" name="windowName" value="myWindow">
Width <input type="text" name="width" value="200">
Height <input type="text" name="height" value="200">
<input type="checkbox" name="directories"> Directories
<input type="checkbox" name="location"> Location
<input type="checkbox" name="menubar"> Menubar
<input type="checkbox" name="scrollbars"> Scrollbars
<input type="checkbox" name="status"> Status
<input type="checkbox" name="toolbar"> Toolbar
<input type="checkbox" name="resizable"> Resizable
<input type="button" value="Create It"
    onClick="createWindow(this.form)">
</form>
```

Which you can try out for yourself:

URL testpage.htm | Name myWindow | Width 200 | Height 200



Managing Your Windows

Maintaining their focus and keeping them centered

You may have noticed that its quite easy to lose pop-up windows - they can become hidden behind the main browser window. This happens when the main browser window regains focus, i.e., becomes the selected window. To avoid this you need to decide whether you want the pop-up window to have exclusive control of the browser, or whether you want it to be closed, or even brought to the front after a short delay.

By placing the following in the document loaded into the pop-up window then the pop-up will remain in front of the main browser window:

```
<body onBlur="window.focus()">
```

This has the side effect on some browsers of inhibiting the use of the main browser window until the pop-up window is closed. If you don't require this feature then you could always refocus the pop-up window after a delay:

```
<body onBlur="setTimeout=('window.focus()',1000)">
```

Which ensures that the pop-up window regains the focus after a delay of one second - enough time to allow the user to interact with the main browser window.

Of course you may decide that if the user has moved the focus to the main browser window, that the pop-up window should be closed:

```
<body onBlur="window.close()">
```

Keeping windows centered

In NN4, the ***screenX*** and ***screenY*** attributes were introduced. In MSIE4 the ***top*** and ***left*** attributes were introduced. By combining the two both NN4 and MSIE4 will allow you to position the window. In earlier browsers the attributes and their values will be safely ignored:

```
window.open('testpage.htm','myExample6','width=200,height=200,
screenX=400,screenY=400,top=400,left=400');
```

Test Example 6

To actually center the pop-up window requires the use of the window objects ***outerWidth*** and ***outerHeight*** properties in NN4 and the screen objects ***width*** and ***height*** properties in MSIE4. However, this results in the pop-up window being centered within the confines of the main browser window in NN4, and centered within the confines of the screen in MSIE4 - not the same thing, unless the main browser window is maximized.

```
<script language="JavaScript"><!--
function centerWindow() {
    if (document.all)
        var xMax = screen.width, yMax = screen.height;
    else
        if (document.layers)
            var xMax = window.outerWidth, yMax = window.outerHeight;
        else
            var xMax = 640, yMax=480;

    var xOffset = (xMax - 200)/2, yOffset = (yMax - 200)/2;

    window.open('testpage.htm','myExample7',
    'width=200,height=200,screenX='+xOffset+',screenY='+yOffset+',
    top='+yOffset+',left='+xOffset+'');
}

centerWindow();
//--></script>
```

Test Example 7

New Window Features

Both NN4 and MSIE4 introduced new window features that can be set when opening up a new window:

NN4.0+	Value	Description
alwaysLowered	boolean	Creates a new window that floats below other windows, whether it is active or not.
alwaysRaised	boolean	Creates a new window that floats on top of other windows, whether it is active or not.
dependent	boolean	Creates a new window as a child of the current window, i.e., it closes when its parent window closes.
hotkeys	boolean	Enables or disables most hotkeys in new window that has no menu bar.
innerHeight	numeric	Specifies the height, in pixels, of the window's content area. Replaces height.
innerWidth	numeric	Specifies the width, in pixels, of the window's content area. Replaces width.
outerHeight	numeric	Specifies the vertical dimension, in pixels, of the outside boundary of the window.
screenX	numeric	Specifies the distance the new window is placed from the left side of the screen.
screenY	numeric	Specifies the distance the new window is placed from the top of the screen.
titlebar	boolean	Controls whether the windows has a title bar.
z-lock	boolean	Controls whether a window rises above other windows when activated.
MSIE4.0+	**Value**	**Description**

fullscreen	boolean	Specifies whether to display the browser in a full-screen or normal window.
channelmode	boolean	Specifies whether to display the window in theater mode and show the channel band.

In NN4, to create a window smaller than 100 pixels high by 100 pixels wide then signed scripts will be required, other features like setting a window as always lowered or offscreen also require signed scripts - this is to avoid a window that the user cannot see.

To find out more about using signed scripts in NN4 see Chapter 7, **JavaScript Security**, in **Netscape's JavaScript Guide**.

Controlling Pop-up Windows

The ***windowHandle*** allows you to control the contents of the pop-up window; either to change the location of the pop-up window, to write HTML code into the window, or to interrogate the window properties.

Changing Location

The following shows how to alter the location of the pop-up window from ***blank.htm*** to testpage.htm:

```
<script language="JavaScript"><!--
myWindow8 = window.open('blank.htm','myExample8','width=200,height=200');
myWindow8.location.href = 'testpage.htm';
//--></script>
```

There are potential problems with the code above. There are occasions where the window may not actually be opened before the change of location is attempted. It is best to change the above code to introduce a slight delay to give the browser a chance to actually open the window first. The following introduces a one second delay before the location is changed:

```
<script language="JavaScript"><!--
myWindow8 = window.open('blank.htm','myExample8','width=200,height=200');
setTimeout("myWindow8.location.href = 'testpage.htm'",1000);
//--></script>
```

Test Example 8

Writing things down

The following demonstrates how to write directly to the pop-up window using the ***windowHandle*** in conjunction with the documents write method. The first occasion that the window is written to will cause the existing contents to be replaced, other following document writes append information to the window.

```
<script language="JavaScript"><!--
myWindow9 = window.open('testpage.htm','myExample9','width=200,height=200');

function update() {
    for (var i=0; i < 10; i++)
        myWindow9.document.write('Message number ' + i + '<br>');
}

setTimeout('update()',1000);
//--></script>
```

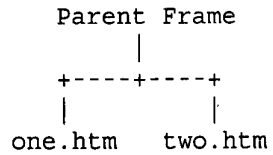
Test Example 9

Interrogation

With frames in a frameset its fairly straight forward to access the parent frame or the child frame, all you need to do is remember the object hierarchy. For example, the following frameset:

```
<frameset cols="50%,50%>
<frame src="one.htm" name="left">
<frame src="two.htm" name="right">
</frameset>
```

Produces the following structure:



To access the parent frame from ***one.htm***, use simply use: ***alert(parent.location.href)***, which will highlight the location of the parent frame, and similarily, to perform the reverse ***alert(window.one.location.href)*** will highlight the location of the left frame from the parent frame, and ***alert(parent.two.location.href)*** will highlight the location of the right frame from the left frame.

Once you've established this simple object syntax, you can use it to read and write almost any property of any other window within the frameset.

However, when you've opened a new window, the window does not form part of a frameset - therefore a new means of accessing the new window from the opener (i.e., the window that opens the new window) is required and vice versa.

Well, accessing the new window is straight forward enough, you just use the window name as shown in the previous example. Modern browsers also include a means of accessing the opener window. Funny enough, by providing an opener property, e.g., ***window.opener***, which will reference the window, if any, that opened the current window. For example ***alert(window.opener.location.href)*** will highlight the location of the current window's opener.

Unfortunately, the opener property was not made available in older browsers - it was introduced in NN3 - but, we are able to create our own window properties/variables at any time:

```
<script language="JavaScript"><!--
myWindow10 = window.open('blank.htm','myExample10','width=200,height=200');
if (newWindow10.opener == null)
    newWindow10.opener = self;
//--></script>
```

Test Example 10

Which checks to see whether the new window already has an opener property, and if not, creates one with a reference to the current window. This new window property can be used as if the browser supplied the opener property itself.

Date pop-up window

To round of this article, I've included a practical example of using pop-up windows. The form button, when pressed, opens up a date selector. Once a date has been selected the date is returned to the form on this page.

Show Calendar

The date selector is described in detail in the article Popup Date Selector

This article has attempted to show you how to create new windows with properties and attributes that you can specify yourself. We have also seen how to write to these windows and how to access one window from another. Hopefully this will encourage you to make more use of the windows ***open()*** method.

View the profile on Martin Webb and the list of other Articles by Martin Webb.

Appendix C

Pop-Ups

POP UP Window

by
William Bontrager, Programmer & Publisher

There are plain popups. And there are popups with altered behavior.

Today you will learn how to make popup windows. And you will learn how to make your popups:

1. Stay on top of any other browser windows, even if the user clicks on another window.
2. Close automatically when the user clicks on another window.
3. Close automatically after 12 seconds have elapsed (or other time interval you specify).

The popups can be closed manually with either a link or a form button, if you decide to provide that functionality.

You will also find out how to launch your popup with either a link, a form button, or when your page loads.

Two kinds of pages are required to make a popup window; (i) the regular page from which the popup launches and (ii) the page that will appear inside the popup window. Both kinds of pages are regular HTML files that you put on your server just like usual.

This article shows you how to make popup windows with three different types of behaviors, which requires three HTML files. However, all three popups can be launched from one regular page.

The demo page at http://willmaster.com/a/6/pl.pl?66demo is a working example of what this article will show you.

Not all browsers are JavaScript compatible, and some browser's compatibility is partial. However, the code presented here should work find with the latest versions of popular browsers.

The Regular Page

The first thing to do is make the regular page:

1. Create a blank HTML page.

   In the HEAD area of your page (between the <head> and </head> tags), put the following five lines of JavaScript code

   --- <script type="text/javascript" language="JavaScript"><!-- function PopUp(url) { window.open (url,"MyPopup",'height=200,width=300'); } //--> </script>

   Notice the height and width numbers. Those designate the size of the popup window in number of pixels.

You may change them.

1.

The function name in the above JavaScript code is PopUp(), with "url" in the parenthesis. That variable name url will know and use the URL you specify when you launch your popup.

2. In the BODY area of your page (between the <body...> and </body> tags), you can put either or both of the following popup launching code --

3. An anchor tag link. This can be a text or image link. This following code demonstrates a text link.

```
<a href="javascript:PopUp('_____')">Make Popup</a>
```

"Make Popup" may be changed to your custom text. Or, it may be replaced with an image.

4. A form button launcher.

```
<form> <input type="button" onClick="PopUp('_____')" value="Make
Popup"> </form>
```

The "Make Popup" button text may be changed.

Repeat the above link and button, three times. That will give you a one set for each of the three popup windows this article shows you how to do.

Save this regular page that you created. A little later on, you will be replace the _____'s with the URL of the popup window pages you are launching.

The Popup Window Page

Now, the next thing to do is to create an HTML file to display in your popup. We'll make a file for a plain popup window. Then we'll give it some behavior.

Create an HTML file for a plain popup window page that can be closed manually with either a link or a button.

```
<html> <body> <a href="javascript:self.close()">click here</a>, <form> <input
type="button" onClick="self.close()" value="click here"> </form> </body> </html>
```

The two instances of "click here" may be changed to reflect your preferred wording. If you prefer a graphic instead of the text link, that is perfectly acceptable.

Save this file for later use. You will be making copies and changing them to provide different behavioral characteristics.

The <body...> tag of the plain popup window page is the key to giving it behavior. And the first behavioral characteristic we'll provide is make it stay as the top browser window until manually closed.

The First Behavioral Characteristic

Make a copy of the plain popup window page you made and save it with file name first.html

Now, put the following line into the body tag:

```
onBlur="self.focus()"
```

The onBlur JavaScript event code tells the browser what to do if the window tries to lose focus, which it does if the user minimizes the popup window or clicks on another window. In our example, the browser will bring the focus back to the popup window.

So long as the focus is on the popup window, it will remain the top browser window.

Something that could happen in the course of your page being viewed, especially if the popup window is loading a large page, is that the popup window will lose focus before it completes loading. In that case, the onBlur code would never be triggered.

To handle that eventuality, also put the following line into the body tag:

```
onLoad="self.focus()"
```

The onLoad JavaScript event code tells the browser what to do as soon as the page has finished loading. In our example, the browser will bring the focus to the popup window. Once loaded and in focus, the onBlur code then can do its job.

So, the popup window page now looks like this:

```
<html> <body onBlur="self.focus()" onLoad="self.focus()"> <a
href="javascript:self.close()">click here</a>, <form> <input
type="button" onClick="self.close()" value="click here"> </form>
</body> </html>
```

Save the file as first.html

Now go to the regular page you created earlier and replace the _____ of one link/button set with: first.html

The set will look something like this:

```
<a href="javascript:PopUp('first.html')">Make Popup</a>
<form>
<input type="button"
onClick="PopUp('first.html')"
value="Make Popup">
</form>
```

Save the regular page and try it out. When you click the link, the popup should appear with the behavior you gave it. If it gives you problems, see how we did it at the demo page: http://willmaster.com/a/6/pl.pl?66demo

The Second Behavioral Characteristic

This time, we'll remove the "must stay in focus" compulsion and replace it with a behavior that causes the popup window to close automatically if it is minimized or if the user clicks

on another window.

Make a copy of the plain popup window page and save it with file name second.html

Put the following line into the body tag:

```
onBlur="self.close()"
```

The onBlur JavaScript event code tells the browser to close the window if it loses focus.

You may also want to add the

```
onLoad="self.focus()"
```

to the body tag to make sure the popup window has focus at the moment it completes loading.

Save the file as: second.html

Now go to the regular page you created earlier and replace the _____'s of another link/button set with: second.html

Save the regular page and try it out.

The Third Behavioral Characteristic

This time, we'll give it a behavior that causes it to close itself automatically after 12 seconds (or any other elapsed time you prefer).

Make a copy of the plain popup window page and save it with file name third.html

Put the following line into the body tag:

```
onLoad="setTimeout('self.close()',12000)"
```

The onLoad JavaScript event code tells the browser to close the window 12000 milliseconds after the page has completed loading. You may change the 12000 to your preferred time period; if you want a full minute, change it to 60000.

Because the onLoad event code is already in use, it can not be used to ensure the window is in focus when it finishes loading. There are ways around it, but it would require coding beyond the scope of this article.

Save the file as: third.html

Now go to the regular page you created earlier and replace the _____'s of another link/button set with: second.html

Save the regular page and try it out.

Instant Popup

If you want your regular page to launch a popup as soon as it has finished loading, add the following to the regular page's <body...> tag:

```
onLoad="PopUp('second.html')"
```

Just replace second.html with the URL of the page you want to appear in the popup window.

Demonstration

The demo page at http://willmaster.com/a/6/pl.pl?66demo provides a visual of how the above works. You'll probably think of many ways that the demonstrated behavior can be used.

Happy, popup windowing!

William Bontrager
Programmer/Publisher, "WillMaster Possibilities" ezine
subscribe-possibilities@willmaster.com
Business Home Page

Back To Top

[Home] [Contact][E-Commerce][FrontPage Index]
[Happy Clients!] [Internet News] [Katsuey]
[Marketing][MacromediaTutorials][Pricing Policies]
[Portfolio][Privacy Statement][Resources A-I]
[Resources J-Z][Search Engines] [Search KatsueyDesignWorks]
[Search Engine Optimization][Services][Table of Contents]
[Tutorial Index][Web Design][WillMaster Index]
[Index WMP Tips][Domain Registration/Web Hosting]